Practical Bloom filter based epidemic forwarding and congestion control in DTNs: A comparative analysis

Ali Marandi, Mahdi Faghi Imani, Kavé Salamatian

Pratical Bloom Filter based Epidemic Forwarding and Congestion Control in DTNs: A comparative analysis

Ali Marandi[a], Mahdi Faghih Imani[b], Kavé Salamatian[c]

[a]*Department of Computer Engineering, Khorasgan (Isfahan) Branch, Islamic Azad University, Isfahan, Iran*
[b]*Department of Computer Engineering Science and Research Branch Islamic Azad University Tehran, Iran*
[c]*University of Savoie, LISTIC, F-74000 Annecy, France*

Abstract

Epidemic forwarding has been proposed as a forwarding technique to achieve opportunistic communication in Delay Tolerant Networks (DTNs). Even if this technique is well known and widely referred, one has to address several practical problems before using it. Unfortunately, while the literature on DTNs is full of new techniques, very little has been done in comparing them. In particular, while Bloom filters have been proposed to exchange information about the buffer content prior to sending information in order to avoid redundant retransmissions, up to our knowledge no real evaluation has been provided to study the tradeoffs that exist for using Bloom Filters in practice. A second practical issue in DTNs is buffer management (resulting from finite buffers) and congestion control (resulting from greedy sources). This has also been the topic of several papers that had already uncovered the difficulty to acquire accurate information mandatory to regulate the data transmission rates and buffer space. In this paper, we fill this gap. We have been implementing a simulation of different proposed congestion control schemes for epidemic forwarding in ns-3 environment. We use this simulation to compare

different proposed schemes and to uncover issues that remain in each one of them. Based on this analysis, we proposed some strategies for Bloom filter management based on windowing and describe implementation tradeoffs. Afterwards, we propose a back-pressure rate control as a well as an aging based buffer managing solution to deal with congestion control. By simulating our proposed mechanisms in ns-3 both with random-waypoint mobility and realistic mobility traces coming from San-Francisco taxicabs, we show that the proposed mechanisms alleviate the challenges of using epidemic forwarding in DTNs.
Keywords: Delay tolerant network, Epidemic forwarding, Bloom filter, Buffer management, Congestion control, Back-pressure

1. Introduction

Delay tolerant networks (DTNs) like Wireless sensor networks, vehicular networks (VANETs), and spontaneous networks are characterized by several major challenges such as intermittent and transient connectivity, volatile links and long delays, that make particular methods and mechanisms mandatory for transmitting over them. In DTN applications, data forwarding follows opportunistic approaches based on store-carry and forward scheme, *i.e.* relay nodes store packets and carry them until an appropriate forwarding opportunity arises. The forwarding decision might be based on the identity of an encountered node when there are information on the likelihood of future contacts, *e.g.* in Prophet [1] where the message with the highest likelihood of being delivered by the encountered node to its final destination is forwarded. However, in several cases such future contacts information are

not available, *e.g.* in very dynamic environments where past cannot be used to predict the future like some VANET scenarios, or even at the initial stage of any DTN scenarios where there is not yet enough historical information to direct the forwarding decision. In these situations, DTN routing techniques like Prophet are not usable and only epidemic forwarding is applicable. This has motivated techniques like spray and wait [2] or spray and focus [3] that begins by a first stage of Epidemic Forwarding limiting the spread of each message to L copies and followed by a stage of waiting or routing to reach the final destination.

"Opportunistic Forwarding" consists of deciding which packet to forward only based on information exchanged between encountering about their buffers' content. "Epidemic Forwarding" [4] is differentiated from other opportunistic forwarding approaches by not making the forwarding decision based on destinations of messages as no information is available about future contact (besides the case where the encountered node is the destination of some packets and these packets are sent in priority). Packets will be eventually delivered to their destinations by epidemic forwarding, since one of carriers will likely encounter the destination.

Indeed the major issue and challenge of epidemic forwarding is to control the redundancy and to avoid useless transmissions. In order to achieve this Vahdat and Becker [4] proposed to exchange between nodes a summary bitmap indicating which packets are already present in encountering nodes buffers. However, this idea is not practical because the nodes need to be informed of an ordered list of all messages circulating in the network in order to interpret the bitmaps. Building and diffusing this list seems impracti-

cable especially in an asynchronous multi-source/multi-destination scenario. Therefore, the proposed summary bitmaps is in fact a list of received packet IDs and its exchange can impose a relatively large overhead. Vahdat and Becker [4] suggested therefore using a *Bloom filter* in order to substantially reduce the space overhead associated with the summary vector. Despite this approach being known from a long time, there is a relatively small number of works that describe practically how to implement Bloom filter based epidemic forwarding and discussed the involved trade-offs. In particular, one needs to deal with defining in a distributed way Bloom filters and how to achieve good transmission overheads vs. transmitted redundancy trade-offs. One of the aims of this paper is to discuss this issue.

In addition, Epidemic Forwarding and more generally DTN opportunistic forwarding schemes have to address some practical challenges in order to be usable. A major issue in all networks is congestion control that happens when the rate of input packets is larger than what the network can accommodate. This issue is more vital in DTNs as packets are likely to stay in buffers for a longer time than in traditional networks. A large part of the literature has assumed unlimited buffers and showed good performance for epidemic forwarding, however the performance is strongly affected by limited buffers, as nodes have to drop packets when their buffers become full. Therefore, congestion control and buffer management are of vital importance in DTNs and influence directly the performance. The issue of congestion control has been studied extensively in the Internet and traditional networks. But due to lack of continuous end-to-end connectivity in DTNs, classical congestion control approaches are not applicable there. Therefore, we need particular

congestion control mechanisms for DTNs that base their decisions only on local information. Another issue is relative to buffer management and deciding what to store in buffers and what to drop. This issue is closely related to indicating packets received at destinations in order to not forward them and to free the space occupied by them. Both buffer management and congestion control have been the subject of different papers each addressing only a small subset of the large set of challenges an epidemic forwarding scheme has to address. Up to our knowledge no comparative analysis of the schemes proposed in the literature, has been implemented, and more globally no one of the previous works have tackled together with the above three challenges: distributed Bloom Filter design, congestion control and buffer management.

In this paper, we will first describe a distributed Bloom filter buffer content exchange scheme that could be used for any opportunistic forwarding scheme (either epidemic or not). We also propose a global framework that integrates most of previous work done on congestion control and buffer management of epidemic forwarding in DTNs. The framework consists of three mechanisms: a back-pressure based injection rate control that controls the rate of injection of new packets from sources in order to ensure that a fast source is not submerging its neighbourhood, a buffer management scheme that discards oldest packets and frees space in buffers following a packet aging similar to [5], and a flow control scheme similar to the one implemented in [6] that prevents congestion by postponing message transfers to congested node until adequate resources are available. Most techniques proposed in the literature can be reduced to particular instantiation of different parameters of the above three steps. We implemented the framework in the *ns-3* simu-

lation environment. This enables us to do a comparative analysis of different proposed schemes over the same scenarios and to analyze their strengths and shortcomings. The comparative analysis is done over both random waypoint mobility, and real mobility traces coming from San Francisco taxis. This comparative analysis will lead us to propose a combination of all three mechanisms that achieve a better performance over the simulated scenarios.

In a nutshell the contributions of this paper are as follow. We first present an in-depth analysis of Bloom Filtered based epidemic forwarding, discuss the alternatives and present practical way of implementing it. Up to our knowledge while proposing Bloom Filter has a long history, no proposition implementable in practice have been provided. This paper fills this gap. A second contribution is relative to the three stage framework, *i.e.*, source injection rate control, buffer management, flow control, proposed for categorizing different DTN congestion control schemes. This framwework provides a taxonomy of the different congestion control schemes that help in understanding the strengths and weaknesses of each proposed scheme. While solutions belonging to each single stage have been proposed, the framework leads to proposing a new congestion control scheme combining elements from existing schemes that cover each of the shortcomings of the individual schemes. The last contribution of this paper is relative to the application of the developped framework to simulate and compare the performance of several of the proposed DTN congestion control schemes. Up to our knowledge no such large scale comparison have been done in the literature. In addition to the trivial interest of making a comparison to know which scheme has the best performance, our comparative analysis provides a marginal benefit analysis

of the different stage of congestion control, showing that this is the node to node flow control that has the largest impact by increasing delivery and reducing strongly packet drop caused by congested buffers.

The rest of this paper is organized as follows. Sec. 2 describes related work. Then we describe in Sec. 3 windowing based Bloom filter management strategies. Next, in Sec. 4 we discuss the importance of congestion control and buffer management and propose some mechanisms to deal with it. Sec. 5 presents our Congestion Control framework. Afterwards, we present a simulation based comparative analysis of different schemes and show how to improve them by combining the three mechanisms. Finally, Sec. 7 concludes the paper.

2. Related Work

Epidemic forwarding was initially proposed in [4]. In order to use better the scarce communication resources in DTNs and to avoid useless transmissions, Vahdat and Becker proposed to exchange between nodes a summary bitmap indicating which packets are already received in nodes. However, as explained before this idea is not practical as building and diffusing an ordered list of all messages circulating in the network seems impracticable especially in an asynchronous multi-source/multi-destination scenario, resulting in exchanging in most implementations of DTNs a list of received packet IDs in place of a bitmap, and imposing a relatively large overhead to exchange this information. Vahdat and Becker [4] already suggested to use a *Bloom filter* to reduce the overhead associated with the summary vector. However despite this approach being known from a long time, there is a relatively small

number of works that describe how to implement and describe the involved trade-offs of designing a distributed Bloom filter exchange scheme in order to exchange buffer content IDs. A thorough survey of the literature shows that the only case that a Bloom Filter implementation is described for buffer content exchange is in the IBR-DTN implementation [7], however the issues are not described. Out of this, all papers just state that "To reduce the size of the summary vector, it may be possible to use compression techniques such as a Bloom Filter", or put the use of Bloom Filter as a future work.

Use of Bloom filter has also been advocated for use in ad-hoc networks however with slightly different aims that exchanging buffer contents. For example, B-SUB (Bloom-filter-based pub-SUB system) [8] is a content-based publish-subscribe system for pocket-switched networks that uses Temporal Counting Bloom Filter (TCBF) to perform content-based networking tasks. Yoneki et al. in [9], proposed another content-based publish/subscribe in mobile ad-hoc networks that uses Bloom filters based aggregated summaries of content subscriptions for building a dynamic event dissemination system and extend ODMRP (On-Demand Multicast Routing Protocol). In [10], Aad et al. a Bloom filter is used to provide anonymity in ad-hoc networks. In [11], Parris et al. designed an Obfuscated Social Network Routing (OSNR) scheme that use Bloom filter data structure to embed the friends list instead of transmitting it in plain text.

Bloom filters have been proposed by [12] to collect the set of relay nodes a packet has crossed from its source to its destination. This collection eases the decision to forward subsequent data packets. IRTF DTN IP Neighbour Discovery (IPND) Internet draft [13] advocated another usage of Bloom Fil-

ters to determine whether a link is bidirectional. In [14] Bloom Filter is used to piggyback in each packet the neighbourhood of the sender at the time of transmission. A relay node use this information to decide to forward a received message only if its neighbourhood is not strictly included into the one of the senders. This approach stops a relay from forwarding messages that are likely to have been received by all neighbours in previous transmissions. However transmission of redundant packets is still possible as a packet might have been received in different neighbourhood, meaning that this scheme does not replace the need for nodes to exchange their buffer contents.

The spray and wait/focus schemes [2, 3] are two hybrid scheme mixing epidemic forwarding at initial stage of the propagation and routing in the second stage. The idea of limiting the number of copies generated in the initial spray phase has a strong congestion control motivation. However, it has not be presented and evaluated as so. As we will see later the congestion *à la* spray and wait can be interpreted inside aging framework.

N-Drop [15] is a congestion control strategy under epidemic routing in DTNs. In this strategy, N is determined as a threshold for buffer size and nf is a counter containing the number of times a packet is forwarded. When a node with a full buffer receives a new packet, it checks whether for any packet $nf > N$. The node removes these packets. If $nf \leq N$ for all packets, the received packet is dropped. In [16], the AFNER congestion control strategy is proposed. In this strategy, each node sorts the list of packets in its buffer in ascending order of their number of forwarding. At each forwarding slot, packets with smaller forwarding numbers are privileged. When a congestion is detected and the buffer becomes larger than a threshold, packets that have

forwarding numbers larger than the average forwarding number needed for packets to reach their destination are assumed to have been delivered to their destinations, and are removed from the buffer. A main issue with AFNER is that finding the average forwarding number needed for packet to reach their destination is not obvious in a decentralized way. Whether N-Drop [15] and AFNER [16] are interesting approaches, they lack some essential components. In particular, this is not only the number of forwarding that controls the interest of storing a packet in the buffer, the number of copies of this packet existing in the neighbourhood is also important. Moreover, without source injection rate control a source can overflow its neighbourhood or send useless packets.

In [17], Grundy et al. proposed, CAFé, a congestion-aware framework for single copy forwarding, that adaptively chooses the next hop based on contact history and statistics, as well as storage statistics. The goal is to distribute the load away from the storage hotspots in order to spread the traffic around. In [18], they extend their approach to multi-copy forwarding and proposed a unified congestion control framework for DTN routing, which encompasses adaptive forwarding and adaptive replication management. However, as stated in the introduction we are interested in this paper in scenarios where a node cannot predict next contacts using history and statistics and has to use epidemic forwarding. Most of the mechanisms proposed in [17] and [18] cannot be applied in such a context. In this paper, we propose congestion control techniques applicable to these challenging cases.

In [19], the authors first describe the looping problem that happens when a node that has removed one packet from its buffer continues to receive it

from other nodes and proposed an Effective Looping Control solution to it. After that they propose a congestion control policy called Credit-based Congestion Control (CCC) that is based on the concept that the oldest as well as most-forwarded messages should be dropped first. In this paper, we use Bloom filters to implement a much more efficient way of dealing with the looping problem than what is proposed in [19]. Moreover, we reuse the aging framework described in [5] that enables a more flexible and easy way to implement buffer management.

Autonomous Congestion Control [20] uses an economic model for a rule-based congestion control mechanism where each router can autonomously decide whether to accept a data packet based on local information such as available storage and the value or risk of accepting the bundle derived from historical statistics. Congestion control is implemented through a backpressure scheme that will inform source nodes of congestion in network by back propagating the denial of storage by an intermediate node through the network. However, this approach is unfortunately not applicable for the scenario of interest in this paper that entails not knowing future contacts and not having a path back to the source to back-propagate the feedback. Nonetheless, the schemes presented in this paper can be considered as autonomous as the decision to forward or drop a packet will only depend on local information.

Thompson et al. [21] developed a congestion control algorithm for intermittently connected networks. In their scheme, a measured congestion level indicator that is derived through information exchanged between nodes about message drops and replications dynamically controls message replication. In [6], a flow control scheme named Buffer Space Advertisement

(BSA) is proposed. It consists of piggybacking in data and keep-alive messages the available free space in node's buffer. This information is used by neighbours to avoid sending messages to nodes whose buffers are almost full. The paper proposes an adaptive buffer space advertisement that uses a congestion coefficient that is, similarly to the Additive Increase Multiplicative Decrease (AIMD) scheme in TCP congestion control, linearly increased when no packet drop happens and multiplicatively decreased when a packet loss happens. The congestion coefficient is used to reduce the total available space of the buffer and therefore to reduce the advertised buffer space. The proposed mechanism maximizes resource utilization by preventing congestion by postponing message transfers until adequate resources are available. We are implementing a similar local flow control scheme in this paper that can be used to implement both the schemes in [21] and [6]. However as shown in [6] the simpler BSA with adaptive buffer space feedback outperformed the congestion control scheme in [21]. For this reason, we only compare our proposed scheme with what developed in [6].

The authors of [22] (SR) proposed for a node that is confronted with congestion to forward newly received packets to non-congested neighbours, *i.e.* neighbours that have still free space in their buffers, to avoid dropping them. Later when the congestion will reduce, the node might retrieve migrated packets. Therefore, congestion control in SR acts as a local routing protocol that diverts messages from their typical routing path and as a response protocol to retrieve them for later forwarding. However, this approach is not applicable in sparse networks where there are no "other" neighbours. Moreover, there is no guarantee that a node can find such non-congested

networks.

Krifa et al. [23] presented different buffer management policies for DTNs showing that the classical drop tail policy is sub-optimal. An optimal buffer management policy either to minimize the average delay or to maximize the average delivery ratio, is proposed that is based on global knowledge about the network. Krifa et al. propose a distributed algorithm that uses statistical learning to approximate the global knowledge. However, the aging framework presented in this paper is more general than the one developed in [23].

The more relevant work to this paper is [5] that describes some of the components reused in this paper. However, besides [5] never being published and having stayed as a technical report, it lacks some major components that have been added in this paper. In particular in this paper, we add a Bloom Filter mechanism enabling nodes to know about their local contents, we do a comparative analysis with other congestion control schemes, and a flow control scheme is introduced.

3. Bloom Filter based Epidemic Forwarding

As explained in the introduction, even if the idea of using a Bloom filter to reduce the overhead of exchanging the buffer contents, goes back to the initial paper on epidemic forwarding [4] and it has been reused frequently in the literature. However, most of the paper just stated a sentence saying that Bloom filters might be used to reduce the size of the exchanged summary vector, or proposing the use of it in future work. As detailed in the previous section, there are rare exceptions like [7] that have implemented Bloom filters for buffer content exchange, however the issues and challenges have not been

described. In this section, we will describe the existing issues and we will propose solutions to solve them.

3.1. Issues and challenges

DTNs are networks that are characterized by the fact that node encounters happen randomly so that finding a deterministic path from the source of information to its destination is not feasible. Whenever one of two encountering nodes is the destination of a packet sitting in the buffer of the other node, a final delivery occurs. If it is not the case, the receiving node acts as a relay and will transport the packet to its final destination or another relay. Because of the dis-connectivity and contact uncertainty, DTN nodes have to transport several packets in their buffers in order to deliver them later.

Basically, DTN routing consists of deciding based on the characteristic of an encountered node and the buffer content which packets to forward to it. In classical routing, the decision about the next node to forward a received packet is made based on the packet's destination, *i.e.* a packet is forwarded to a node that is in the path for reaching the final destination based on the routing table. However, in DTNs the situation is more complex, as the nodes mobility is uncertain and the node does not know precisely its upcoming contacts. In situations where a node cannot decide (because of lack of information or because of high dynamicity of the network) about the likelihood of an encountered node delivering its forwarded message, the identity of the encountered node is not anymore useful for the forwarding decision. In such situation, Epidemic Forwarding is used. In Epidemic Forwarding, whenever two nodes encounter they make the relaying decision without considering the packets destination (besides when the encountered node is the

final destination of the packet) but only based on their buffer content. In particular, when two nodes become in contact they should avoid sending packets that the other node has already received.

The above description defines four main challenges and issues for Epidemic forwarding in DTNs:

1. **Nodes' Buffer Comparison**: in order to avoid exchanging redundant packets (packets that the encountered node has already received), the two encountering nodes have to inform each other of the content of their buffers. In their seminal paper, Vahdat and Becker [4] proposed to use a summary bitmap indicating which packets are already received in nodes and to exchange this summary between nodes. However, this idea is not applicable in practice as the nodes need to have access to an ordered list containing all messages circulating in the network in order to build and interpret the bitmaps. Exchanging this summary list can impose a relatively large overhead especially when the contact time between nodes is short. For this reason, [4] suggested to use a *Bloom filter* in order to substantially reduce the space overhead associated with the summary vector.
2. **Reception Acknowledgment**: Buffer space and connectivity bandwidth, *i.e.*, the available capacity of transmission during a contact between nodes, are the major resources that have to be used efficiently in DTNs. A trivial way to improve the utilization of these two important resources is to inform nodes in the network about the packets that have reached their destinations. This information enables nodes to free the space allocated to these packets and ensure that they do not waste the

scarce connectivity bandwidth by being re-forwarded. For achieving this purpose, we need to implement a feedback scheme that will signal the reception of packets. This scheme will also be implemented in form of Bloom Filters to reduce its burden.

3. **Neighbourhood Management**: because of mobility, the neighbourhood of each node becomes very dynamic. However, the information forwarding strongly depends on the neighbourhood of the node and the knowledge that each node has about the content in the buffer of its neighbours. Neighbourhood management has therefore a strong impact on the performance of DTN schemes.
4. **Buffer Management**: as explained before, nodes in DTN scenarios have to store a large number of packets in their buffers. However, the buffer space is limited and a node will have to decide which packet to store in its buffer. While this question is essential for congestion control in DTNs and will be discussed later. Nonetheless, Bloom filters have important impacts on Buffer management and *vice-versa*. We will develop later on this last point.

All the above four issues will be addressed in this paper. In the next sections, we will describe the system architecture, the Bloom filter management and in particular three strategies that will address different issues.

3.2. System architecture

Let us assume that each packet injected in the network is identified by a combination of a uniquely assigned 16 bits source node ID, and a 16 bits serial index assigned locally by the source, *i.e.*, each packet is identified by a

32 bits packet ID, we will name pktID, that uniquely identifies each packet in the DTN and can be generated asynchronously in each node.

We assume that each node in the DTN maintains a buffer with a capacity of N packets that the node will forward. In addition to this buffer, several lists are maintained. For all schemes at least two lists are maintained: a *neighbours* list and a *destReceived* list. The first list contains the information related to nodes that are known to be in the neighbourhood of the node. In particular, it will contain for each neighbour the last Bloom filters provided by it. The second list contains the set of packets that are known to have been received at destinations. This last list is used for signalling the reception of packets at their destinations and enables nodes to remove from their buffer packets received at destinations.

3.3. Neighbourhood management

As explained before, neighbourhood management is a fundamental issue of DTNs. In particular, a fundamental question is how to decide if two nodes are in reach of each other. In this paper, we will assume that there is no particular support from lower layers (in form of synchronization or loss of synchronization) and neighbourhood detection and management is only done through message exchange. Whenever node A receives a message from node B, it is in its neighbourhood and can receive messages sent from A. A node is removed from neighbourhood if no packet is received from it during a disconnection timer duration. We moreover assume that nodes send periodically (with a period less than the disconnection timer) a beacon message to keep connections and neighbourhood alive.

The above scheme has two main limitations. First, the delay between the

time when two nodes are physically able to exchange information and the time when they figure out that they are neighbours can be as high as the inter-beacon interval. This delay can be a major problem especially when mobiles are fast moving as the contact duration might not be enough to figure out that two nodes are in contact. Increasing beacon frequency can reduce this effect. However, in sparse networks most of the beacons are never received, as connectivity is scarce. Increasing the rate of beacon transmission can result in beacon transmission pumping most of node battery.

The second issue is the dual problem. When the neighbour of a node moves and goes out of its reach, it needs a time equal to disconnection timer delay to decide that the node is not anymore in the neighbourhood. This means that a node might still continue to send messages to a node that is not anymore in its neighbourhood. This last point results in a large number of unsuccessful transmissions that consumes node energy without having any benefit. The solution to this issue might be to reduce the disconnection delay, but as the disconnection delay should be larger than the beacon transmission interval, this means reducing the later and leading to the same issue as described before. This shows the importance of the beacon delay for the performance of neighbour management and the efficiency of the Epidemic Forwarding scheme. Indeed a desirable mechanisms would be a lower layer mechanism running at the physical layer and detecting the presence or absence of a neighbour, but we will not assume such scheme in this paper.

3.4. Bloom filter management

We explained before that each packet in the network is identified uniquely by a 4 bytes Packet ID. Now let us assume that a node has in its buffer L

packets and it wants to inform another encountered node about its buffer content. As described before, several solutions for this problem have been proposed in the literature, among which Bloom filter [24]. Bloom filter is a data structure representing in a very compact way, set membership. Therefore, the Bloom filter can be used to represent the content of the buffer in a node. For this purpose, the node generates a Bloom filter that contains L ($L \leq N$) packet IDs and sends the information needed to retrieve this Bloom filter to the encountered node that use these information to check the membership of its local packets and to decide which new packet to send back. The information needed to retrieve the Bloom filter consists of two components: information about the K hashing functions that are needed for the operation of the Bloom filter, and the membership vector that contains the bits set by the hashing functions. For the first component, we will assume that the node ID is used as the seed of a random generator that is used to generate random hash functions. Meaning that knowing the node ID enables to retrieve the hash functions used by this node. The membership vector is sent directly to the neighbours.

Bloom filters performance is controlled by the false alarm probability, *i.e.* the probability that an entry that is not in the set defined by the Bloom filter is falsely reported as being in the set. This probability can be derived as a function of the number of hash functions K and the length of the membership vector M through a well-known relationship [24]. Generally, one chooses K in order to minimize the probability of false alarms. With this choice, if one wishes to insert N values in the Bloom filter and set a target false alarm probability p, the minimal length of the membership vector M and the needed

number of hash functions that can achieve this false alarm rate is given as:

$$\begin{cases} M & = -\frac{N \ln p}{(\ln 2)^2} \\ K & = \frac{M}{N} \ln 2 \end{cases} \quad (1)$$

For example, if one wants to insert 50 pktIDs in its buffer in a Bloom filter and achieve a false alarm probability less than 2% (one packet per Bloom filter), one will need $M = 407.11$ bits and $K = 5.6$ hash functions. By rounding these values, and aligning the memory to byte units, one needs 51 bytes to transfer the Bloom filter to neighbours in place of the 200 bytes needed to transfer 50 packet ID of 4 bytes each. The above Eq. shows that the memory requirements of Bloom filter increases with the number of inserted values. One has to limit the number of inserted values or to increase the acceptable false alarm rate in order to control the size of the Bloom filter.

Nonetheless, over the time, the number of packets in the buffer of a node increases and therefore the number of values that have to be inserted in the Bloom filters increases, resulting in an linearly (with N) increasing Bloom filter size or an increasing false alarm probability if the size is held constant. We have therefore to use a sliding window in order to manage the Bloom filter contents. A major trade-off results between sliding window length (and the Bloom filter size) and the probability of retransmitting a redundant packet (resulting from a Bloom filter false alarm or from an active packet being dropped out of the window). This trade-off represents the buffer management challenge and has major performance impact on epidemic forwarding. In the forthcoming, we will describe two different strategies to deal with the above trade-off.

3.4.1. Strategy A: Simple Bloom Filter

The first and trivial strategy, we will call strategy A, consists of choosing a maximal number of inserted values N (N can be larger than L the node buffer size to account for packet that are destined to the node and have been removed from the buffer) and an acceptable false error probability p and designing and simply sending in each packet a Bloom filter containing the pktIDs of last N packets received by the node. The receiving node uses these Bloom filters to detect which packets have been received by the neighbour in order to not forward them. Moreover, by checking the destination node ID of a packet with the reception status of the neighbour indicated by its Bloom filter, one can detect if a packet has reached its destination and to remove it from its buffer. In this case, the trade-off is between the size N (or equivalently the size of the Bloom filter) and the probability of transmitting a redundant packet.

3.4.2. Strategy B : Differential Encoding

The second strategy, named B, consists of extending the idea of differential encoding to Bloom filter. Differential encoding has been widely used in compression and consists of first encoding completely an item and to send thereafter only the difference of the fully encoded item with its followers. This achieves a larger compression ratio as the differences can be encoded with much less bits than a full encoding. One can periodically fully re-encode an item in order to resynchronize. Extension of this idea to Bloom filters is straightforward. We assume that each beacon sends periodically by the node to announce its presence, contains a "**big Bloom filter**", named bBF, containing N entries, while each packet send by the node piggybacks a "**small**

Bloom filter", named sBF, containing the $n << N$ last entries. A node checking the reception of a packet by a neighbour will first check the small Bloom filter and check thereafter the big one if the first check is negative. This strategy achieves a better trade off than strategy A, but at the cost of an eventual loss of synchronization that results from beacons being send in larger interval times.

3.4.3. Strategy C: Adaptive Differential Encoding

The third strategy is an extension of Differential Encoding that puts in the sBF piggybacked on each data packet, only packets that have been added from the time of generation of the previous beacon. This enables to decide on the size of the forwarded sBF at the time of transmission and adapt the parameters it in order to minimize the overhead size. Clearly Strategy C should achieves the least overhead but is the more prone to loss of synchronization issues.

3.5. Explicit Reception Notification

Buffer space and connectivity bandwidth are the major resources that have to be used efficiently in DTNs. A trivial way to improve the utilization of these resources is to inform nodes in the network about the packets that have reached their destinations. This information enables nodes to free the space allocated to these packets and ensure that they do not waste the scarce connectivity bandwidth by being re-forwarded. For achieving this purpose, we need to implement a Explicit Reception Notification (ERN) feedback scheme that will signal the reception of packets. This has already be pointed out in [25] where the IMMUNE and VACCINE approaches are developed for

communication among instrumented whales. [25] propose to use an "anti-packet" containing an explicit notification of the reception of a message by its final destination and exchanging it between encountering nodes.

In this paper we adapt a similar idea to the context of using piggybacked Bloom filters. We implement the "anti-packet"" feedback by adding into beacons a Bloom filter, named the explicit Bloom Filter, eBF, containing a list of J last packets received by its destination. Whenever a node receives an eBF Bloom filter it checks all packets in its buffer and adds the packet matched by the eBF into a local "destReceived" list before removing them from its buffer. This list is used thereafter to generate the eBF Bloom filter of that node that can be inserted in each beacon along with for example the bBF of Strategy B that contained packets in the buffer. The eBF plays an important role as it acts as a collective acknowledge propagating in the network and freeing buffer space occupied by packets that have been delivered (implement something similar to the VACCINE scheme in [25] however with larger scope as the overhead of the Bloom Filter is lower than explicitly exchange delivery notifications). We will assume in the forthcoming that the eBF is implemented in Buffers, *i.e.*, the beacon in Strategy B consists of two components: a bBF with a maximal size limit L, and an eBF containing a list of the J last packets received by its destination. More details about the performance of this scheme and the trade-off involved are presented in the forthcoming.

3.6. Forwarding scheme

We assume that the node maintains for each "active" neighbour a data structure containing the set of Bloom filters provided by the neighbour (the

sBF, bBF and eBF filters), a list of PktID named *notReceivedYet* list containing the IDs of packets in the node buffers but not yet received at this neighbour, and information about the last packet forwarded to this neighbour. When a node has an opportunity (given by the scheduler or resulting from a contact) to send a packet to a neighbour, it first checks (using the last Bloom filters received from the neighbour) among the packet received from the time the last packet was forwarded to the neighbour which one are not received at the neighbour and adds them to the *notReceivedYet* list. This last list enables a node to simply avoid forwarding duplicate packets to their neighbours by only sending packets from their *notReceivedYet* lists. Among packets in this list, packets that have as destination the neighbour have priority and will be forwarded first. Otherwise, a randomly selected packet from the *notReceivedYet* is forwarded to the neighbour. A forwarded packet is removed from the *notReceivedYet* list. To ensure synchronization and to manage packets not received by the destination, the *notReceivedYet* is fully reconstructed (by checking if any packet in the buffer is not received by the neighbour) whenever a beacon is received.

The scheduling between neighbours is also simply done by first checking if anyone of them is the final destination of a packet in the buffer. If it is the case the node is given priority. If no final destination exists among neighbours, one node is chose at random and packets are forwarded to it. The priority given to final destination node decreases the average delivery delay. However, provided no destination exists in the neighbourhood, we choose one of the neighbours randomly and send a randomly selected packet to it to avoid discrimination. The sequence number of the sent packet should be removed

from *notReceivedYet* of the receiver. This scheduling policy can be refined to accommodate for congestion situations, *e.g.*, in case on congestion packets likely to be discarded from buffer are getting higher priority in order to eventually find place in neighbours (hot potatoes scheduling used for example in [22]). We will use such a congestion-aware scheduling as explained in sec. 6.1.4.

The forwarding scheme presented here can be adapted to accommodate any specific DTN routing scheme by choosing a particular packet to forward in the *notReceivedYet* list in place of a random one like in Epidemic Forwarding. This shows that the Bloom Filter management system presented in this paper is not only relevant to Epidemic Forwarding but also to other DTN routing schemes like Prophet [1], *etc.* However, in this paper we will focus on its use in Epidemic Forwarding schemes.

4. Congestion Control and Buffer management

Congestion control is a major issue both in DTNs and connected networks. As DTNs are intermittently connected, packets remain for a long time in nodes' buffers waiting for a transmission opportunity. Congestion appears in DTNs when buffers are full and nodes have to drop incoming packets, not playing anymore the role of relay. Moreover, as these packets are not put in buffers, the neighbors keep in retransmitting them resulting in severe waste of bandwidth. Therefore, a practical system using epidemic forwarding needs a congestion control mechanism. The congestion control consists of at least three components: a source injection rate control mechanism that ensures that a greedy packet source do not fill the buffers both

locally and in the neighborhood, a buffer management mechanism that wisely manages which packet should stay in the buffer and decides which should be removed, and a flow control mechanism that uses information about neighbor buffer occupancy to decide to forward packets or refrain from it. Because of its major importance and impact on DTN performance, Congestion Control has been investigated thoroughly in the literature as described in the Related Works section. In this section, we will present a framework inspired by what described in [5] regrouping most of the existing work on congestion Control and Buffer management.

4.1. Buffer management

As explained before, congestion happens in DTNs by buffers becoming full. Buffer management aiming into deciding which packet should stay in the buffer and which should be removed is a fundamental part of congestion control. Generic buffer management can be implemented using the "aging" concept introduced in [5]. The idea is to assign to each packet an age that can depends on several parameters, like the time from packet generation, the number of time the packet was forwarded, the number of hop the packet has travelled, or even socio-economical parameters like the monetary cost of storing or forwarding a packet, *etc.* The age represents the benefit of storing a packet in buffer, the smaller it is the more valuable is the packet. Aging framework offers a very flexible framework that encompasses several other buffer management proposals in the literature like [2, 3, 15, 16, 18, 19]. All these schemes can be re-expressed as aging schemes with different aging functions. Moreover, as suggested in [5] the TTL field in the IP header is a very good target for transporting the age of a packet. For this purpose, let us

assume that the maximal age of a packet is 255. Instead of only decrementing the TTL at each hop as is traditionally done in networks, one can decrement or increment this field to represent a quantified remaining Time To Live for the packet. At packet creation time, the TTL field is set to 255, and every time it is forwarded it is, its TTL is set to 255-AGE, meaning that the TTL is decremented following an aging function. Whenever the TTL hits zero, the packet is erased from buffers and not forwarded anymore. In this paper we have used the below described aging function:

- In order to control the spread of the packets in the network, every time a packet is going to be forwarded, its age in buffer is incremented by a fixed amount K_0, and the TTL of the forwarded packet is set accordingly. Therefore, K_0 plays the role of hop count and limit the number of time a packet is forwarded by a node.

- In order to control the number of copy of a packet in the DTN, we use a homeostatic approach; whenever a node encounters a node that has received the packet, it increases by K_1 the age of it in its buffer, reducing the lifetime of the packet. This will implement the limitation to a limited number of copies of any message as proposed for example in [2, 3].

- We also need to control the lifetime of packets in the network. However the amount of time a packet can live in a buffer depends on the contact statistics, *i.e.* if the network is sparse and the contacts not frequent, packets should live longer in order to have the opportunity to be delivered, while they should live less when the contacts are more

frequent. Therefore, we use each packet reception as a clock tick and increase the age of all packets in a node's buffer by K_2.

- Nevertheless, the role of buffer management is to deal with situation when the buffer becomes full and a new packet should be accommodated. In this case, an incoming packet that is younger (has a larger TTL) than the oldest packet in buffer, replaces it. We call this the "oldest discard" scheme. However in order to avoid the looping effect described in [19], even if a packet is erased from the buffer (or is not put in buffer because it is was too old), its ID should still be announced in the next Bloom Filters. For Strategy B, we will just do this for the sBF.

Based on the above described mechanisms, if packets arrive on average each Δ time units, one can expect the lifetime of a packet in a the network to be at most $\min\{\frac{255}{K_2}\Delta, N\Delta\}$. Roughly if $\frac{255}{K_2} < N$ aging controls the message lifetime, otherwise this is the buffer size that controls the lifetime of a packet. Nonetheless, packets have to be alive enough time to ensure that the network will find the opportunity to find a path from packet source to its destination to deliver it. This gives a rule of thumb for setting K_2. If we want to ensure that packet lifetime in the network does not go beyond $T_{\max}$, one should control the packets' lifetime through K_2 and set $K_2 < \frac{255}{T_{\max}\Delta}$. However this is valid if the buffer never gets full. Therefore, it is important to avoid packet drops resulting from full buffer. The last mechanism of congestion control achieves this.

4.2. Flow control

We need to avoid packet drops resulting from full buffers. For this purpose we can, similarly to the flow control mechanism proposed in [6], upon a node's buffer becoming full, ask the neighbors to refrain from sending new packets. This can achieved by explicitly piggybacking into the header of send packets (both data packets and beacons) a field containing the available space in the node buffer. This field will be used by neighbors to refrain from sending new packets that can lead to killing packets before natural death by age. However this heuristic has a side-effect; aging efficiency is directly related to the number of piggy backed Bloom filters that each node receives, and the network time scale is controlled by the rate of packet arrivals. In order to not disturb the aging process, nodes that refrain from sending data payloads still continue to send Bloom filters as well as buffer capacity. Although this flow control idea largely decreases the number of packet drops, some packet drops are still observed due to concurrent transmissions from different nodes to each node and therefore the "oldest discard" scheme is still needed. In addition to the above mechanisms, the ERN described earlier plays also an important role in buffer management.

4.3. Source injection rate control

Another major component of congestion control component is the control of injection rate of new packets by the sources. An uncontrolled source can submerge its own node's and neighbors' buffer, contributing strongly to congestion and creating fairness issues. We therefore need a way of controlling the source injection rate, such that it is not too high and overflow the network, or too low and starving the network from new packets. The rule of thumb

here is to ensure that the rate of injection of new packets do not exceed the capacity of the network to transfer these packets.

We need to precise the meaning of capacity here. When N packets coexist in a node's buffer, the competition between these packets to access the communication capacity, when another node is encountered, results for each packet in an inter-node capacity of $\frac{1}{N}$ of the communication capacity. The same issue happening in all hop of the communication, its meaning that the capacity depend on the intermediate node buffer occupation. Overall the available communication capacity for a single packet from source to destination is the sum over all possible paths from source to destination of the minimal inter-node capacity between all nodes in the paths. Unfortunately, this value is impossible to derive for a node, in particular for DTNs without permanent connectivity. We therefore need a heuristic that can, using only local information, at least gives an upper bound for this capacity. The idea is that the capacity to forward a packet out of the neighborhood of its source is an upper bound of the overall available capacity for forwarding a packet. In other terms, the upper bound on the capacity is obtained by trying to infer the capacity of transmission in its two hop neighborhood. This capacity is inferred by this heuristic: let us assume that a node generated a packet at time T_0, and the node saw later at time $T_0 + \Delta$ the same packet reported by the Bloom Filter of an encountered node that he has not itself given him the packet, the source node can conclude that the overall capacity for forwarding this packet is at least one packet per interval Δ. Indeed, this heuristic is very rough, and it can even be seen as too restrictive as the delay between the reception of a packet by a remote node and the time the source node sees it

can be relatively long. However it is hard to be improved taking into account the very low information that a DTN gives about upstream capacity.

The injection rate control is implemented by a token bucket rate control scheme using a list named *waitingList* that contains up to σ pktIDs generated by the source. The source can inject new packets in the node buffer only if there is empty space in the *waitingList*. The injection rate is controlled by removing a packetID from *waitingList* whenever one of the below events happen: following the above described heuristic, whenever a source S receives a Bloom filter that indicates that a pktID in the *waitingList* has been received by node A, and the source has not forwarded itself the packet with that pktID to A ; when the source comes in direct contact with the destination of a packet it has generated. The sum of the occurrence rates of the two above events acts as an upper bound to the capacity of the network. This source injection rate control scheme acts as a back pressure mechanism that limits the rate of injection of new packets in the network. However its performance is highly dependent on the mobility characteristics and on the sparsity of a network. We will evaluate its performance later in the paper.

5. Congestion Control framework

As we mentioned before, we propose a framework that implements most of the existing work on congestion control and Buffer management. Now, we add more details about the implementation of each mechanism.

N-Drop scheme [15] will forward a packet at most N time before dropping it. N-Drop can be emulated in our framework by initializing K_1, and K_2 to zero, and setting $K_0 = \frac{255}{N}$. Through this choice a packet is forwarded at

most N time before reaching a TTL=0.

AFNER scheme [16], is implemented simply by setting, similarly to N-Drop, K_1, and K_2 to zero, and setting $K_0 = \frac{255}{\bar{N}}$, where $\bar{N}$ is the mean number of forwarding needed to reach the destination. We also need a slight change in the scheduler that has to forward the youngest packet in the buffer rather than choosing it randomly.

The Buffer Space Advertisement (BSA) is implemented through space advertisement in the beacon and packet headers, with setting K_0, K_1, K_2 to 0. The advertised free buffer space is derived using the Additive Increase, Multiplicative Decrease scheme described and using the same set of parameters that suggested in [6]. We name this implementation BSA-AIMD.

Storage Routing (SR)[22] is also implemented in our proposed framework, by giving, in case of congestion, higher priority in the scheduler to older packets in the buffer and to forward them to nodes with buffer capacity. However, in order to know about available space in neighbours, we have implemented in SR a Buffer Space Advertisement piggybacked in packet headers. Meaning that the SR implementation is also using a Flow control based on buffer space announcement that is not proposed in the initial SR proposition. This means that we can expect the performance of our SR implementation to be better than the one proposed initially in [22].

6. Performance Evaluation

In order to evaluate the performance of the proposed practical Bloom filter based epidemic forwarding methods and congestion control mechanisms, we implemented in the ns-3 simulation environment the proposed schemes.

We are indeed aware of the shortage of simulation relative to representativeness and generalization. However, the simulation can show weaknesses and needs indeed to be comforted with real deployment. In order to evaluate the proposed schemes we have used random waypoint mobility as well as the mobility resulting from San Francisco taxi traces [26]. We are aware of the shortcoming of simulation based on random waypoint mobility described in [27] and we will therefore not draw our conclusions only based on these simulations but are evaluating also on more realistic traces like the San Francisco Taxi traces.

6.1. Random Waypoint mobility evaluation

6.1.1. Neighbourhood Management

Let us first evaluate the impact of neighbour management we described in section 3.3. For this purpose we simulated 40 nodes moving in a 1000×1000 grid with a WiFi transmission range of 50 unit each, *i.e.* around 30% of the mobility area is covered by connectivity, with a rate of 1 Mbps. Each node implements the neighbourhood strategy, described earlier with beacons broadcast and disconnection delay. We set the disconnection delay as 10% larger than the beacon interval. We have used strategy A and 50 bytes of Bloom filters. We show in Fig. 1 the number of forwarded and received packets for different beacon delays ranging from 0.1 sec to 5 sec. The gap between the two curves is the number of packets forwarded but that are not received by any node. We show in Fig. 1, the connectivity and efficiency. Connectivity is defined as the ratio of the number of packets and beacons received to the overall number of packets and beacons sent. Efficiency is accounting the energy efficiency, *i.e.*, it evaluates the energy cost of the neighbour man-

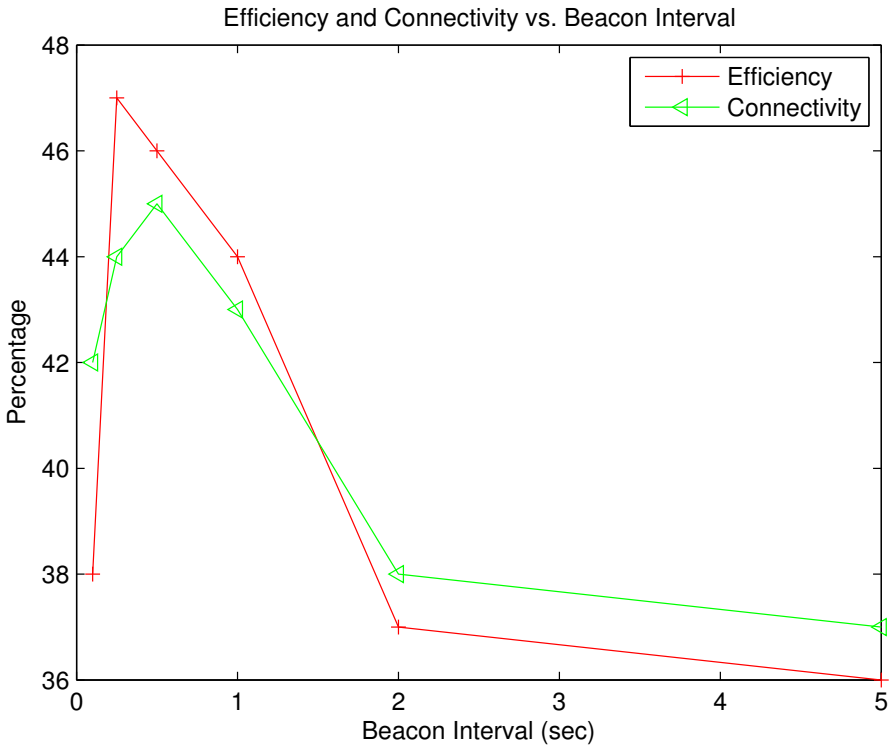

Figure 1: Impact of beacon delay on number of forwarded and received packets

agement and packet transmission vs. its benefits as data packet received at nodes. To account for the smaller size of beacon compared to data packet, 50 bytes vs. 1000 bytes, we have accounted the energy cost of beacon packet as 5% of data packets. Interestingly the both the connectivity and efficiency curve first increase to attain a peak a decreases thereafter. This behaviour comes from the fact that a short beacon interval results in a large overhead of beacon transmission while a too long beacon interval results in loosing transmission opportunity. It can be observed that at best the efficiency is 47% for beacon interval 0.2 sec. This low efficiency is explained by the fact the connectivity is also very low (less than 45%). The connectivity and efficiency drop fast to 36% with larger beacon intervals. Based on the above curve we decided to set in the forthcoming a beacon delay of 0.5 sec that maximize the connectivity and is at 1% of the maximum of efficiency.

6.1.2. Bloom Filter Management

We thereafter evaluated on the same simulation scenario (40 nodes moving in a 1000×1000 grid with a WiFi transmission range of 50 unit), a scenario where each node acts simultaneously as source, relay and destination. The sources are assumed to be greedy meaning that whenever there is an opportunity to insert a new packet in the network they have data to send and insert it. We moreover assume that each packet has 1000 bytes of payload. The destination of messages is chosen randomly among all 40 nodes in the network. The buffer size of nodes is assumed to be 50 packets. After a node is detected to be in the neighbourhood, packets are sent to it directly (no broadcast is used for data packets). No other congestion management scheme is applied. All Bloom filters are designed with a target false alarm

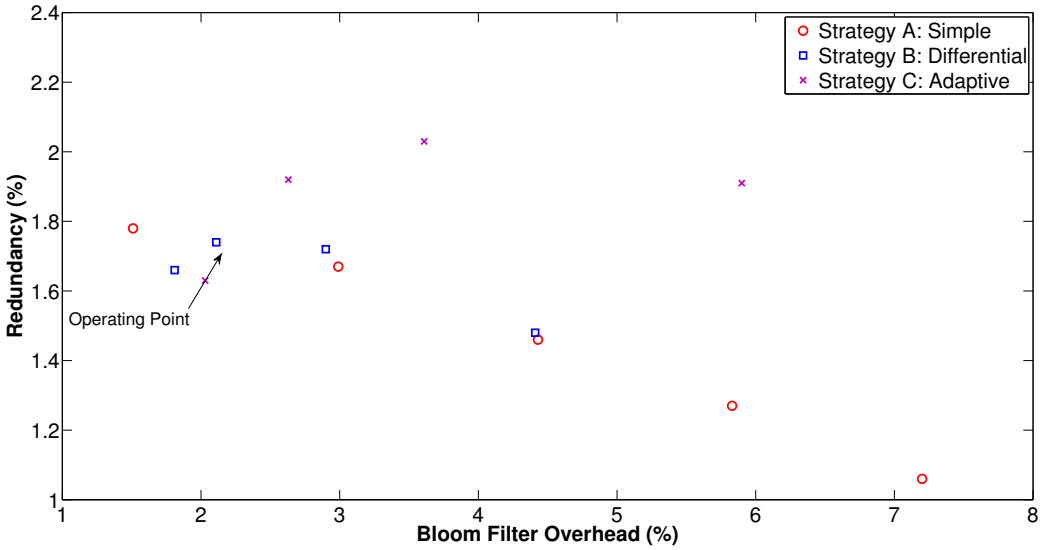

Figure 2: Comparison of redundancy vs. Bloom Filter overhead with different Bloom filter sizes for several Bloom filter management strategies

probability of 2%. We show in Figure 2 the evaluation of the redundancy, defined as the proportion of redundant packets received versus the overhead, that is measured as the proportion of received bytes that are dedicated to Bloom filters (both in sBF and bBF). This is measured for the three Bloom filter management strategies. The simple strategy A is measured for Bloom filter size of 10, 20, 50, 70 and 120. The Differential encoding strategy B is implemented with bBF size of 50 (so that all packet in buffer are reported) and sBF size ranging from 10 to 30 (each data packet contains from 10 to 30 last packet received). The adaptive strategy C is implemented for bBF size ranging from 30 to 120 and sBF size calculated adaptively. The curve shows that as expected the larger is the Bloom Filter used, the larger the overhead becomes and the smaller the redundancy. This is very visible in particular for the strategy A, where the redundancy decreases constantly with larger

overhead. However the Differential encoding strategy B has also a similar behaviour with almost same overhead vs. redundancy behaviour. It would have been expected that the trade-off should be more favourable for strategy B. However as explained strategy B suffers from loss of synchronization that happens every time a beacon is lost. This effect is more visible in Adaptive Strategy C, where the overall redundancy is larger than strategy A and B because of inserting in sBFs only packets received till the last beacon transmission. Figure 2 shows the efficiency of using the Bloom filters in place of summary reports. With less than 2% of overhead the redundancy falls below 2%. Based on these results we decided to use throughout the paper the strategy B, with bBF=50 and sBF=12 that achieves a redundancy 1.71% with an overhead of 2.11% (indicated as operating point in Fig. 2). The choice of strategy B in place of strategy A was motivated by the fact that for each data packet transmission in strategy A, we have to insert up to 50 packets in the piggybacked BF, while in strategy B this is reduced to 12 packets resulting in almost division by four of the processing time for each data packets transmission for the same performance.

6.1.3. Explicit Reception Notification scheme

We validate in this section the ERN Bloom Filters based schemes described in section 3.5. We implemented an ERN scheme putting the pktID of up to 150 packets that are known to be received in beacons and broadcasting them to neighbours. We show in Fig. 3 the evolution of the number of packets received at destination with two hypotheses: without ERN and with ERN. In this simulation, no other Congestion Control or Buffer management schemes are used beyond ERN. The curve shows clearly the saturation ef-

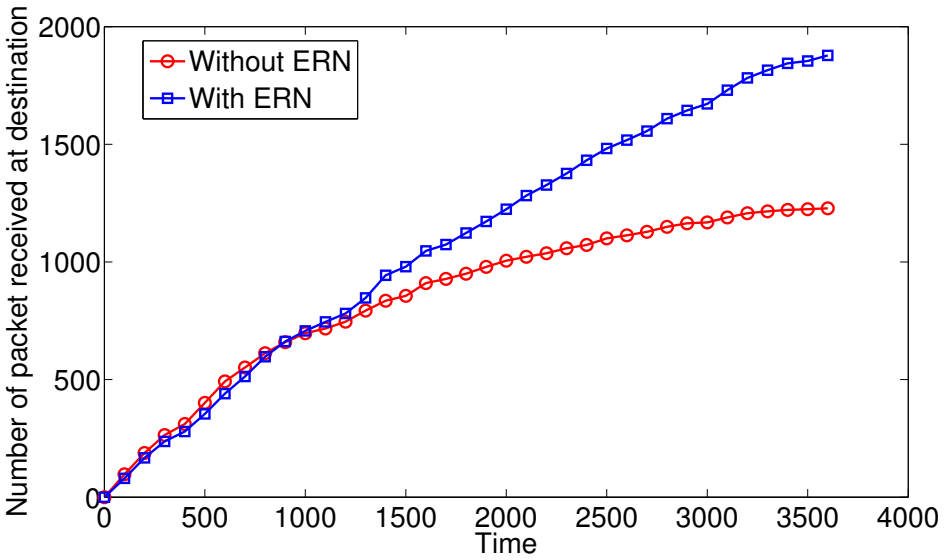

Figure 3: Comparison of number of packet received at destination over time with Explicit Reception Notification and without it.

fect of congestion. After around 1300 sec the network get saturated and the number of packets reaching their destination without ERN reaches a plateau, while the usage of ERN frees space in buffers and enables a sustained packet delivery over the network. At the end of the simulation, the number of packets received with ERN is almost doubled compared to the use of ERN.

6.1.4. Congestion Control

In this section, we will evaluate the different congestion control propositions and in particular the mechanisms proposed in this paper. As explained before, we have implemented the aging-based Buffer Management, the flow control and the injection rate control described in section 4. By setting the value of different parameters, we have implemented 5 concurrent congestion control propositions, namely Drop-N, AFNER, SR, and BSA-

AIMD, and compared them with the different mechanisms presented in this paper. In order to understand the marginal impact of each mechanism, we have implemented them incrementally, *i.e.*, we first implement a DTN system with Aging based Buffer management (denoted by Ag) by setting $K_0 = 25, K_1 = 10, K2 = 0.25$; it is improved by adding a flow control based on Buffer Space announcement piggybacked in packet and beacons (system Ag+FB); finally a source injection control based on Back Pressure is added resulting in a Ag+FB+BP. This means that we are comparing 8 epidemic forwarding schemes. All these implementations take advantage of the Bloom Filter management strategy B described earlier with sBF containing up to 12 packets and bBF containing up to 50 packets as well as the ERN scheme. Indeed, these schemes were not implemented in the original propositions of Drop-N, AFNER, SR, and BSA-AIMD, meaning that the provided performance can be expected to be better that what originally proposed. All the simulation are run over 10 execution with different random seed and the average along with 2 standard deviation confidence interval are presented.

As we have several schemes to compare, in order to simplify the comparison we have divided the set of concurrent schemes into two subsets: buffer management alone schemes consisting of Drop-N, AFNER and Ag scheme, and Flow Control based schemes consisting of BSA, SR, Ag+FB, Ag+FB+BP. We show in Fig. 4 the performance in term of number of packets received at final destinations, as a function of time for the different concurrent congestion control schemes. As can be seen, all curves are almost linearly increasing, meaning an almost constant goodput. However, there is a clear separation between schemes using only buffer management and spread

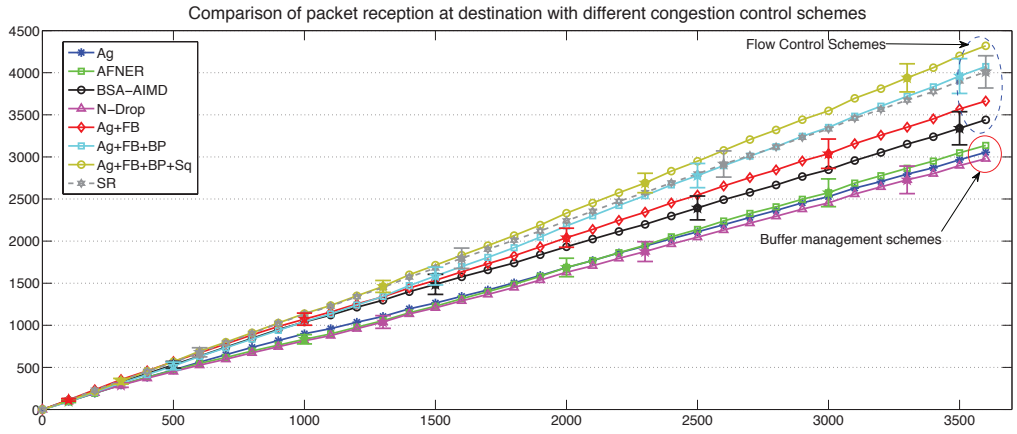

Figure 4: Comparison of number of packets received at destinations per unit of time for congestion control mechanisms for Random Waypoint mobility

control, Drop-N, AFNER and Aging, with schemes, like SR, BSA-AIMD, Ag+FB and Ag+FB+BP, adding Flow control in form of buffer space announcement. The goodput in the second group of techniques is around 30% larger than in the first group. Interestingly, as can be seen from confidence interval (that are shown for different shifted value to ease the reading of the figure), the difference in the three buffer management alone scheme are not statistically significant at least after one hour of simulation. However the flow control schemes show more separated performance as observed by almost non crossing confidence intervals. In order to have a better understanding, we plot in Fig. 5 the evolution of the number of drops observed in the same set of schemes. A packet drop happens when a packet is received by a node but it cannot put it in its buffer because of lack of space and has to drop it, *e.g.* using Oldest Discard approach, a packet drop happens only when the received packet is older than all packets in the buffer. Fig. 5 shows that with buffer management techniques, drops happen very frequently with

10-15% of send packets being dropped. However, adding Flow control in form of Buffer Space Advertisement, reduces strongly the packet drops. In particular, BSA-AIMD almost removes all drops, but cannot achieve a good delivery to final destinations. A more precise investigation shows that while the AIMD scheme has succeed in avoiding the buffer to become full, it has also decreased the exchange between node and reduced the diversity of packets in buffers, resulting in a lower delivery performance. Interestingly the SR scheme achieves better delivery performance than the BSA-AIMD scheme. This can be explained by observing that SR adds to the BSA flow control a specific scheduling that consists of giving higher priority to older packets when the buffers are full. This means that in place of just discarding a packet when it becomes the oldest packet in a buffer, SR scheme tries to forward it to a neighbour with available capacity. However, the drop observed when using SR is much larger than other Flow Control schemes. This can be explained by the fact that an old packet forwarded to node's neighbours is more likely to be older than all packets in the receiver and being dropped. In comparison with the BSA-AIMD and SR schemes, the Ag+FB+BP scheme presented in this paper achieves a better delivery performance with relatively low drops. The interest of the Back Pressure injection rate control is also visible in the graph as the delivery is increased by 15% by adding the Back Pressure without increasing the drops (the curve for drops in Ag+FB and Ag+FB+BP are almost superposed so we do not show the curve of Ag+FB in Fig. 5). However the good performance of SR techniques motivates us to investigate if in addition to Aging, Flow control and Back Pressure described earlier, one could benefit from the clever scheduling used in SR, *i.e.*, giving old packet

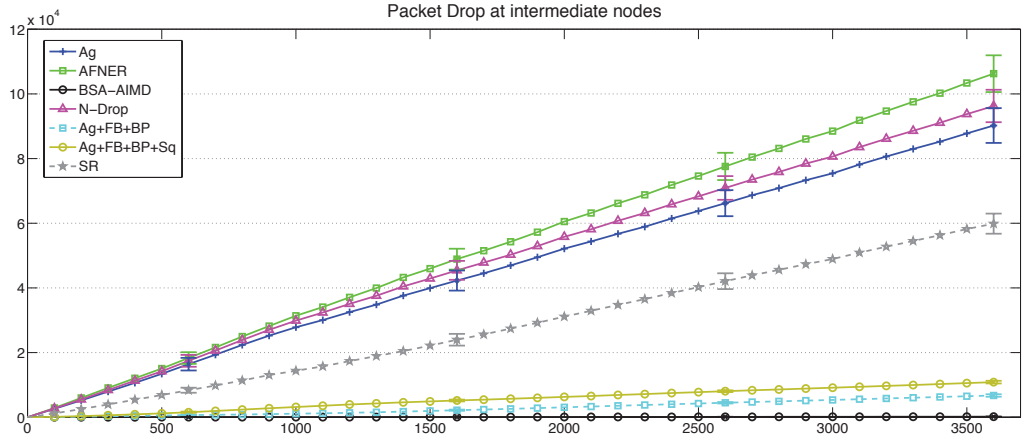

Figure 5: Comparison of number of packet drops per unit of time for congestion control mechanisms for Random Waypoint mobility

more forwarding priority. For this purpose we added to the Ag+FB+BP scheme a SR type of Scheduling and named it a Ag+FB+BP+Sq. As can be seen from 4, this scheme achieves the best delivery performance at the cost of a slightly higher drop rate (compared to Ag+FB+BP) that results from the older packets to be more likely to be dropped at reception than younger packets. Comparing the curve of packet drop of SR and Ag+FB+BP+Sq gives also some more insights. The packet drops are strongly reduced by the use of Aging that reduces the buffer occupation and therefore the likelihood of need for packet drop in neighbours. For all curves we have also provided confidence intervals in order to evaluate the statistical significance of the results.

We show in table 1 the overall performance in terms of delivery ratio, mean delay of packets received at destinations (measured in minutes), and mean buffer size. As can be seen the delivery ratio increases strongly from

Table 1: Overall performance comparison for the Random Waypoint Mobility

Scheme	Delivery Ratio	Mean Delay (min)	Mean Buffer
N-Drop	13.04%	15.54	49
OD	13.83 %	11.4	49
Aging	14.56 %	4.3	49
AFNER	14.73 %	4.23	49
BSA-AIMD	16.31 %	3.73	49
Ag+FB	17.39 %	3.22	49
SR	26.24 %	2.99	49
Ag+FB+BP	26.85 %	2.98	48
Ag+FB+BP+Sq	45%	2.85	48

13% for the worst scheme to 45% for the Ag+FB+BP+Sq that achieves the best performance and the mean delay also decreases from 15.5 to 2.85 mins.

The above evaluation shows that all the four components described in Sec.4: buffer management, flow control, scheduling and injection rate control have a strong impact on the final delivery performance. All in all, the schemes proposed in this paper, succeed for a random waypoint mobility model in reducing strongly the congestion (evaluated by packet drops), while improving the packet delivery to final destinations by 20% compared to BSA-AIMD that was the previously best-known congestion control scheme.

6.2. San Francisco Trace evaluation

The random waypoint gave some insights about the performance of the different strategies and congestion control schemes. However as explained

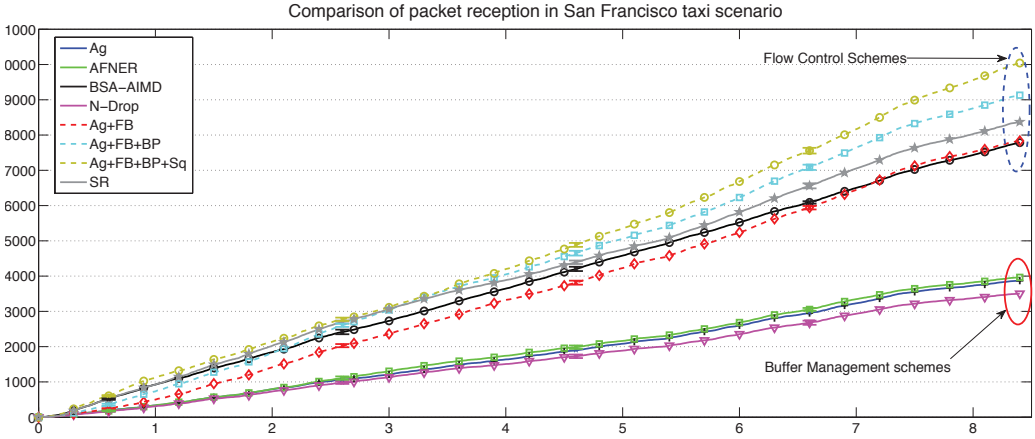

Figure 6: Comparison of number of packets received at destinations per unit of time for congestion control mechanisms for the Taxi trace

before, random waypoint simulations are plagued with convergence issues as described in [27]. In order to evaluate the different congestion control schemes in a more realistic mobility model, we used traces coming from San Francisco GPS dataset. This dataset contains GPS coordinates of approximately 500 taxis collected over 30 days in the San Francisco Bay Area [26]. We assumed that the taxis are instrumented with WIFI receivers with a range of 100 meters and we only used 100 taxis among the 500 in the dataset. This scenario is sparser than the previous random waypoint one and we had to run it for a longer time, one day or equivalently 84 000 seconds. We show in Fig. 6 the results in term of number of packets received at their final destinations for different strategies. The results obtained in Fig.6 are very similar to what observed in Fig.4, with the same ranking between the different schemes. We show in table 2 the performance attained by the different schemes. The achieved delivery ratios are less than what attained in Random

Waypoint. This can be explained by the high sparseness of the network. The achieved mean delay decreases from 256 to 66.9 that is relatively large but understandable with regards to the sparsity of the network.

A more precise analysis of the Fig.7 shows that the Flow control plays well its role and reduces the packet drop by 92% (from 161,187 to only 1511 for Ag+FB+BP+Sq scheme) resulting in a notable increase of the number of received packets. Sparser, resulting in lower opportunity for packet exchange and that the value of K_2 is relatively smaller, resulting in a lower impact of the aging mechanism. Another noteworthy observation from Fig.6 is relative to the fact that the number of received packets for the scenario with source control rate is initially less than for case without the source control rate. However source control rate catches up later. This can also be explained by the sparsity of this scenario. At the beginning, the source injection rate control reduces the number of new packets that can be injected into the network, resulting in a lower number of circulating packets in the network and therefore a smaller number of packet delivery, however with time the buffers become full and the role of source injection rate control shows itself. The analysis of the San Francisco scenario illustrates also that all mechanisms of congestion control are needed and have a considerable impact on performance. Interestingly in this scenario the confidence intervals are smaller (they can hardly be seen). This is explained comparitively with the random waypoint case that the mobility in this scenario is not random while the previous scenarios also had random way point mobility that is know to induced large variations.

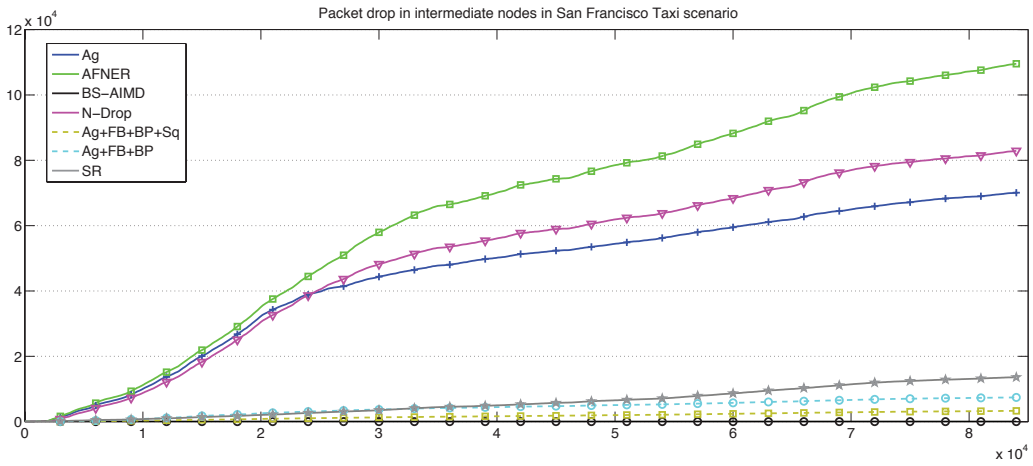

Figure 7: Comparison of number of packets drop for different congestion control mechanisms over the Taxi trace

Table 2: Overall performance comparison for the San Francisco Taxies Mobility

Scheme	Delivery Ratio	Mean Delay (min)	Mean Buffer
N-Drop	5.52%	256.58	44
OD	6.43 %	192.26	43
Aging	7.2 %	140.99	43
AFNER	7.36 %	134.17	43
BSA-AIMD	10.12 %	85.93	42
Ag+FB	15.87 %	74.93	42
SR	16.61 %	72.43	41
Ag+FB+BP	18.12 %	71.38	41
Ag+FB+BP+Sq	22.9%	66.95	41

7. Conclusions

We described in this paper how to use in practice Bloom Filters to exchange buffer content in epidemic forwarding schemes. We proposed three Bloom Filter strategies : a simple, a differential and an adpative strategy and compared their performance in term of the redundancy to overhead tradeoff. We also proposed an explicit Reception Notification scheme using Bloom Filters send back in keep-alive beacons. We moreover described the issue of congestion control in DTNs by showing that it can be decomposed into four components: buffer management, flow control, source injection rate control and scheduling. We thereafter compared 4 existing congestion control schemes : N-Drop, AFNER, SR and BSA-AIMD with an incremental implementation of the different scheme proposed in the paper. The comparison was done using a ns-3 simulation of all the concurrent schemes. We showed that the presented congestion control mechanisms improve the performance incrementally and improve over the best known congestion control schemes.

The major conclusions of this paper are as follow.First, by comparing several possible Bloom filter design, we proposed a practical based on a differential coding that is using both short Bloom Filter piggybacked in each sent packet and big Bloom Filter send only in periodic beacons. This scheme ensure that by adding an overhead of only 2% the likelihood of sending to a node a message it has already received become less than 2%. We also proposed a three staged congestion control framework and showed through analysis and simulation validation that among these three stages the flow control based on Buffer Space announcement piggy backed in packet and beacons ensuring that the rate of packet transmission to congestionned node

is reduced, has the largest marginal impact, followed by buffer management and finally source injection rate management. The combination of these three mechanisms attain 30% higher goodput than any of the alternative schemes previously proposed with 1/4 to 1/5 of their packet drops. Last but not least the main conclusion of the paper is to provide a practical solution for deployment of epidemic forwarding based DTN solutions that addresses the major challenges of highly dynamic network environments where DTN routing cannot be implemented.

References

[1] A. Lindgren, A. Doria, O. Schelén, Probabilistic routing in intermittently connected networks, ACM SIGMOBILE Mobile Computing and Communications Review 7 (3) (2003) 19–20.

[2] T. Spyropoulos, K. Psounis, C. S. Raghavendra, Spray and wait: an efficient routing scheme for intermittently connected mobile networks, in: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, ACM, 2005, pp. 252–259.

[3] T. Spyropoulos, K. Psounis, C. S. Raghavendra, Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility, in: Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on, IEEE, 2007, pp. 79–85.

[4] A. Vahdat, D. Becker, et al., Epidemic routing for partially connected ad

hoc networks, Tech. rep., Technical Report CS-200006, Duke University (2000).

[5] A. El Fawal, J. Le Boudec, K. Salamatian, Performance analysis of self limiting epidemic forwarding, Tech. rep., Technical Report LCA-REPORT-2006-127, EPFL (2006).

[6] J. Lakkakorpi, M. Pitkänen, J. Ott, Using buffer space advertisements to avoid congestion in mobile opportunistic DTNs, Wired/Wireless Internet Communications (2011) 386–397.

[7] S. Schildt, J. Morgenroth, W.-B. Pottner, L. Wolf, IBR-DTN: A lightweight, modular and highly portable bundle protocol implementation, in: Electronic Communications of the EASST, 2011.

[8] Y. Zhao, J. Wu, B-sub: A practical bloom-filter-based publish-subscribe system for human networks, in: Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on, IEEE, 2010, pp. 634–643.

[9] E. Yoneki, J. Bacon, An adaptive approach to content-based subscription in mobile ad hoc networks, in: Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on, IEEE, 2004, pp. 92–97.

[10] I. Aad, C. Castelluccia, J. Huubaux, Packet coding for strong anonymity in ad hoc networks, in: Securecomm and Workshops, 2006, IEEE, 2006, pp. 1–10.

[11] I. Parris, T. Henderson, Privacy-enhanced social-network routing, Computer Communications 35 (1) (2012) 62–74.

[12] M. Särelä, J. Ott, J. Ylitalo, Fast inter-domain mobility with in-packet bloom filters, in: Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture, MobiArch '10, ACM, New York, NY, USA, 2010.

[13] D. Ellard, R. Altmann, A. Gladd, D. Brown, Dtn ip neighbor discovery (ipnd) internet draft (2012).
URL `http://tools.ietf.org/html/draft-irtf-dtnrg-ipnd-02`

[14] F. Angius, M. Gerla, G. Pau, Bloogo: Bloom filter based gossip algorithm for wireless ndn, in: Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications, NoM '12, ACM, New York, NY, USA, 2012, pp. 25–30.

[15] Y. Li, L. Zhao, Z. Liu, Q. Liu, N-drop: congestion control strategy under epidemic routing in DTN, in: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly, ACM, 2009, pp. 457–460.

[16] L. Yun, C. Xinjian, L. Qilie, Y. Xiaohu, A novel congestion control strategy in delay tolerant networks, in: Future Networks, 2010. ICFN'10. Second International Conference on, IEEE, 2010, pp. 233–237.

[17] A. Grundy, M. Radenkovic, Promoting congestion control in oppor-

tunistic networks, the Proceedings of IEEE WiMob 2010, Niagara Falls, Canada, pp 324-330.

[18] M. Radenkovic, A. Grundy, Framework for utility driven congestion control in delay tolerant opportunistic networks, in: IWCMC, 2011, pp. 448–454.

[19] L. Leela-Amornsin, H. Esaki, Heuristic congestion control for message deletion in delay tolerant network, Smart Spaces and Next Generation Wired/Wireless Networking (2010) 287–298.

[20] S. Burleigh, E. Jennings, J. Schoolcraft, Autonomous congestion control in delay-tolerant networks.

[21] N. Thompson, S. C. Nelson, M. Bakht, T. F. Abdelzaher, R. Kravets, Retiring replicants: Congestion control for intermittently-connected networks, in: INFOCOM, 2010, pp. 1118–1126.

[22] M. Seligman, K. Fall, P. Mundur, Storage routing for DTN congestion control, Wireless communications and mobile computing 7 (10) (2007) 1183–1196.

[23] A. Krifa, C. Barakat, T. Spyropoulos, Message drop and scheduling in DTNs: Theory and practice, IEEE Trans. Mob. Comput. 11 (9) (2012) 1470–1483.

[24] B. H. Bloom, Space/time trade-offs in hash coding with allowable errors, Commun. ACM 13 (7) (1970) 422–426.

[25] Z. J. Haas, T. Small, A new networking model for biological applications of ad hoc sensor networks, Networking, IEEE/ACM Transactions on 14 (1) (2006) 27–40.

[26] M. Piórkowski, N. Sarafijanovic-Djukic, M. Grossglauser, A parsimonious model of mobile partitioned networks with clustering, in: Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International, IEEE, 2009, pp. 1–10.

[27] J. Le Boudec, M. Vojnovic, Perfect simulation and stationarity of a class of mobility models, in: INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, Vol. 4, Ieee, 2005, pp. 2743–2754.